Credit Card Fraud Detection

Ali R Kaya

Contents

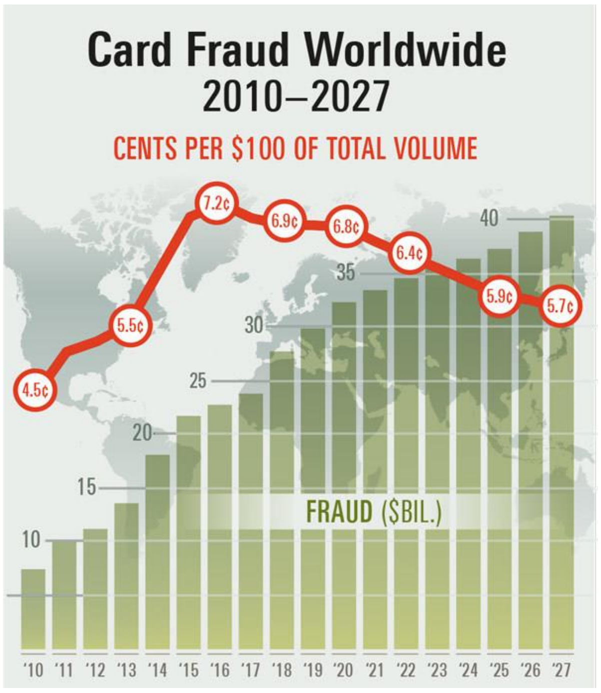
List of Figures

List of Tables

1. Why Fraud Detection?

Card fraud can be defined as the action of making purchases via credit cards, debit cards etc. without being authorized. It is a daily life issue which causes a great number of people to suffer and a great amount of resources to be lost. It can happen to anyone in our society since our transactions are mostly done by either credit cards or debit cards.

Figure 1: Credit Card Fraud in US $ Bil. 2010-2027 (Estimated)



Source: The Nilson Report, November 2019

As seen from the above figure, the loss from the card fraud is expected to reach 40 Billion $, which means that every 5.7 cents of 100 $ spent either by a credit card or a debit card will belong to a fraudulent transaction, by 2027. As seen from the figure, the amount of fraudulent transactions per 100$ reached its peak in 2016. However, the amount of fraudulent transactions in billion $ continues to increase without any interruption. This story tells us that the rate of increase in the amount of genuine transactions is greater than the rate of increase in the amount of fraudulent transactions. However, these figures only account for the financial aspect of fraudulent transactions which do not account for the time spent to resolve the issues and the harm given to the well-being of the society.

In this project, we aim to detect a fraudulent transaction and to prevent it from happening so that it is possible to decrease the amount of resources wasted and to improve the quality of life for all parties.

2. About the Data

The dataset is acquired from Kaggle. Originally, the data was published by the collaboration of Worldline the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles).

There are 284.407 credit card transactions, 492 are labeled as fraudulent and 284.315 are labeled as genuine. In total, 1.825 transactions (27 of them are labeled as fraudulent) have transaction amount of

0$. Those transactions may not be true fraudulent transactions. They may be conducted by the credit card or debit card owner but denied for any reason. In those cases, it is hard to verify that the transactions are fraudulent since the card owner may not contact to the bank regarding the situation but may make the transactions with another card. For this reason, those transactions are dropped from the study.

The dataset consists of 31 features which are V1-V28, Time, Amount and Class. The features V1-V28 are acquired as a result of Principle Component Analysis. For this reason, as we will see below, there are no correlation among those features. Time feature indicates the seconds passed from the beginning of the records. In order to consolidate two-day transactions, we generated another feature, Hour, which is based on Time feature to see how the transactions distributed in hours. Amount variable is the amounts of the transactions. Since the data are gathered in Europe, it is possible to assume that the currency of the transactions is Euro. Finally, the Class variable indicates if a transaction is genuine or fraudulent.

Since there are only two categories of transactions in the dataset, it is a binary classification problem. In a binary classification problem, it is convenient to represent the classes with 0 and 1. The negative class gets the label of 0 and the positive class gets the label of 1. In a classification problem, the negative class represents the class which has the greater number of observations. In this study, the genuine transactions are represented by negative class and the fraudulent transactions are represented by positive class.

2.1. Imbalanced Data

Imbalanced data refers to the case where classes are not represented equally in a data set. In other words, the class distributions are biased or skewed towards one class. The ideal balanced data set should have equal representation of each class. For example, in a binary class data set, in which there are only two classes, each class should consist of 50% of all observations. This pattern can be applied to multi-class data sets. If the pattern is violated, then we have an imbalanced data set.

It is possible to confuse the concepts of imbalanced classification and unbalanced classification. While the former refers to "a classification predicting modelling problem where the number of examples across the classes is not equal" and the latter refers to the deterioration of the balanced structure of the data set. For this reason, using two concepts interchangeably will lead to misunderstanding of the research problem (Brownlee, 2020).

In the credit card dataset, the classes are extremely imbalanced. The ratio of fraudulent transactions to genuine transactions is 1:578. In other words, there are 578 genuine transactions for each fraudulent transaction. It is possible to express the class distribution with percentage of each class in the dataset. While the genuine transactions account for 99.83% and the fraudulent transactions only account for 0.17% of all transactions.

The reasons, in general, for an imbalanced data are:

1. Problems in the data acquisition process
2. The nature of the data

In our case, the imbalanced structure of the data set is the result of the industry standard. The fraudulent transactions are expected to be a tiny proportion of all credit card transactions. It is possible to observe the same pattern in the distribution of classes for claim detection, spam detection, default prediction, churn prediction etc.

It is possible to face with issues with an imbalanced dataset. Because the machine learning algorithms tend to perform better with balanced datasets. It is possible to observe this pattern by generating a synthetic balanced dataset and compare the results. However, imbalanced data set does not cause problems all the time. We will discuss this idea briefly in the next section.

2.2. Why Balanced Data Matters?

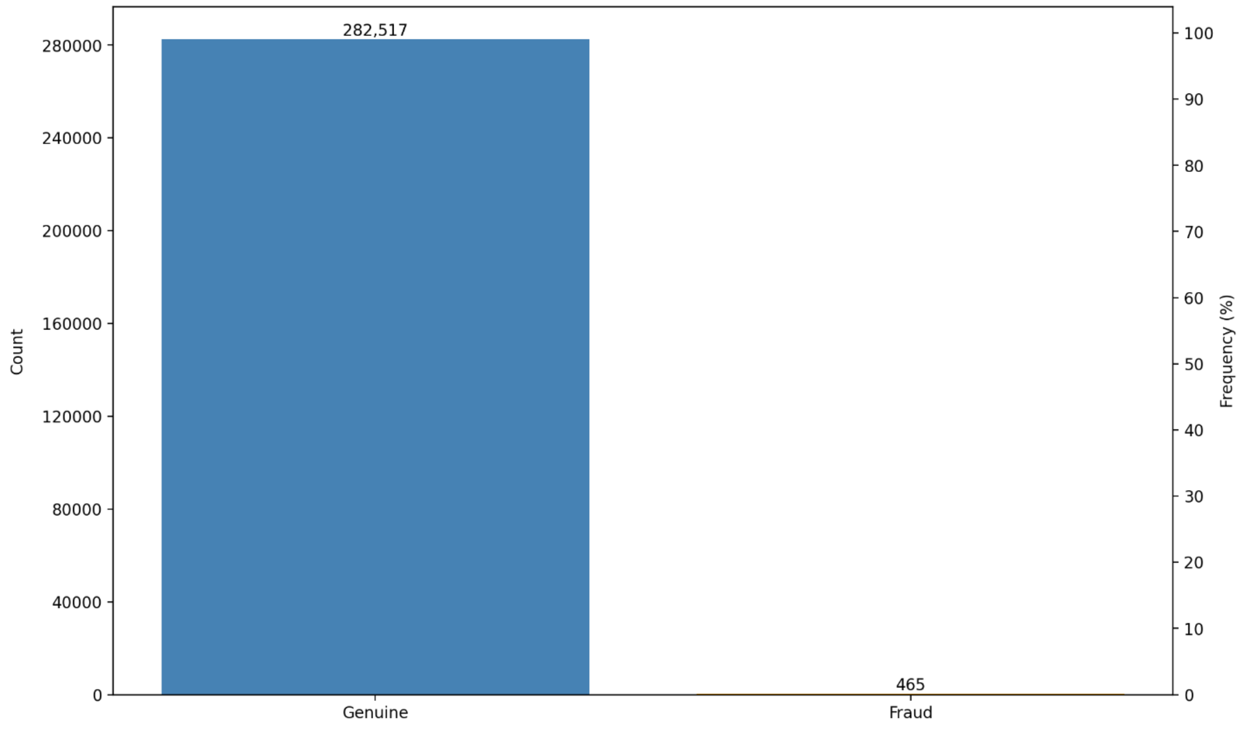
In an imbalanced data set the classes are unequally distributed. As discussed in this StackExchange post, the problem with imbalanced datasets does not arise from the difference in the number of observations for each class but the lack of enough observations in the minority class for a pattern to be recognized by the machine learning algorithm. For this reason, having a *slightly* imbalanced dataset may not cause any problem; unless, there is enough information retained for the minority class(es).

According to Brownlee (2020) 'most machine learning algorithms for classification predictive models are designed and demonstrated on problems that assume an equal distribution of classes.' For this reason, having an imbalanced data set will let the minority class disguised in among the majority class and not be able to be detected effectively. For this reason, in an imbalanced data set, the focus will be on the minority class. For example, in the credit card fraud detection dataset, the aim of the machine learning algorithm is to detect each and every fraudulent transaction. Since, a false positive practically does not cost any resources to the financial institution. On the other hand, a false negative, which can be described as a naive behavior, will definitely result in loss of valuables. For a rather technical explanation of imbalanced class distribution, an interested reader can refer to Rocca (2019).

3. Exploratory Data Analysis

The class distribution of transactions can be seen in Figure 2. Also, in the Figure 3, the heatmap which show the correlation among features by color-coded cells is presented. The feature V1-V28 are acquired through PCA so that they are orthogonal to each other which means that there is no correlation among those features (input variables). This pattern can be clearly seen in the heatmap.

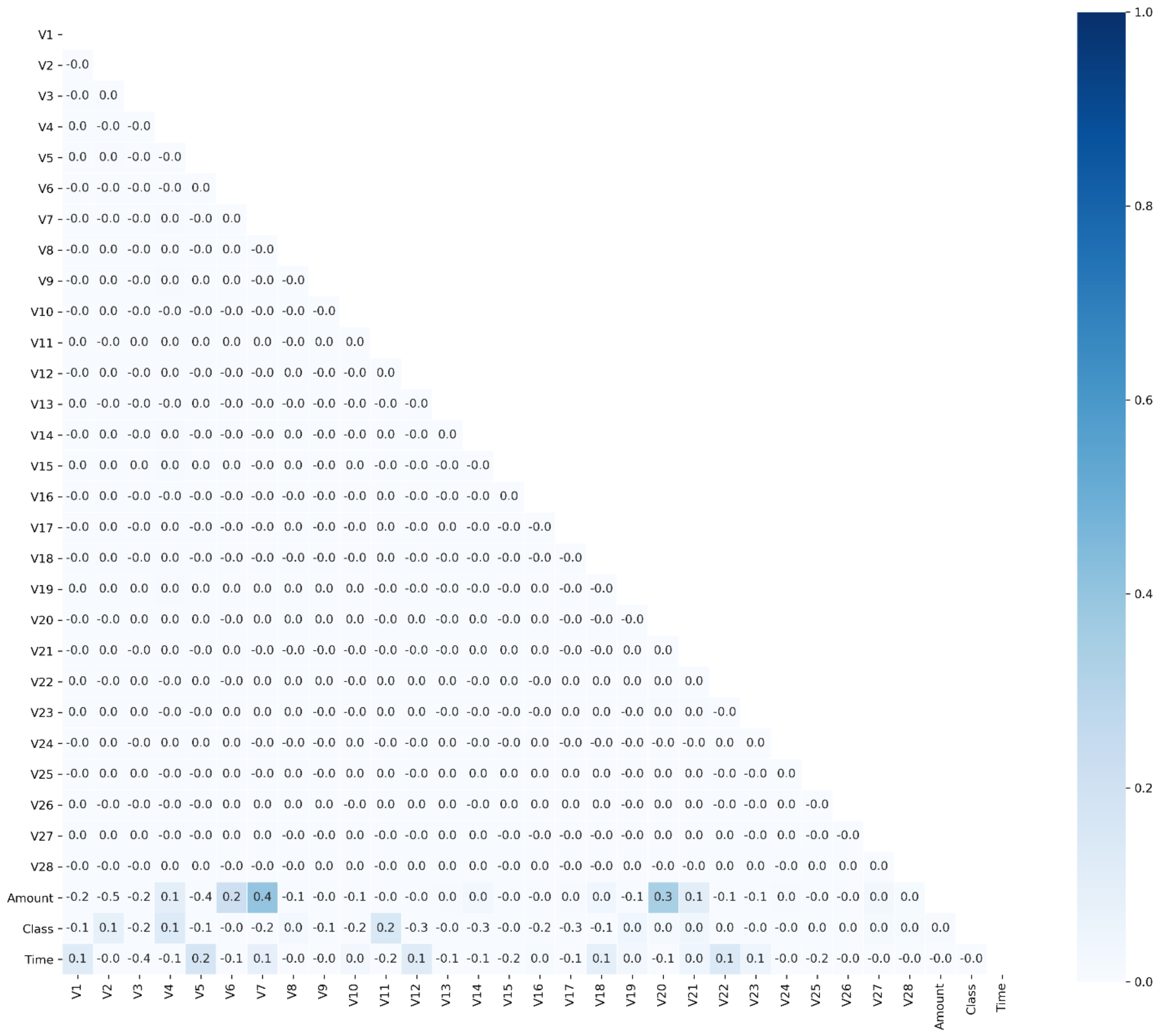
Figure 2: Number of Transactions by Class



The correlation coefficients for V7-Amount and V20-Amount relations are the highest among all others. For this reason, it may be a good idea to have a closer look to those relationships. The scatter plots are plotted in Figure 4.

In the below, we have two scatter plots to show the relationship between 'Amount'-'V7' (on the left) and 'Amount'-'V20' (on the right). Even though, the correlation coefficients tell us that there are some correlations between those pairs, we see that the correlation coefficients are mostly inflated because of the outliers. As a result, there is no linear relationship between those pairs of variables.

Figure 3: Heatmap of Features with Correlation Coefficients



Because the correlation analysis is not fruitful in our case, it is best to move with bivariate analysis. Thus, we will explore the distribution of each feature (‘V1-28’) by target variable. The motivation is to find the features that do not have similar distributions. For this reason, we split the dataset into two, by class, and plotted the distribution of each feature, for two of the classes, in the same plot. We were able to identify 8 features (V9, V10, V11, V12, V14, V16, V17 and V18). However, the limitation of having a small dataset prevents us from only selecting those variables instead we will proceed without discriminating any variable.

In order to give a tangible example, we will provide the distributions of ‘V17’, as an example of distinguishing features, and ‘V22’, as an example of not being able to distinguish features.

Figure 4: Scatter Plots of V7-Amount and V20-Amount

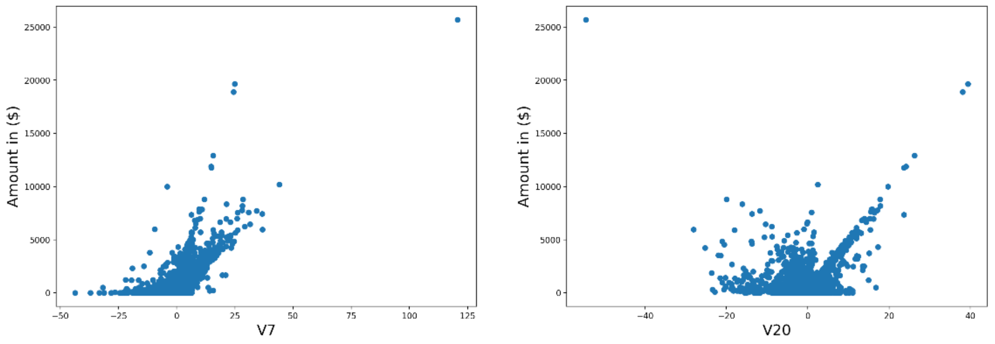
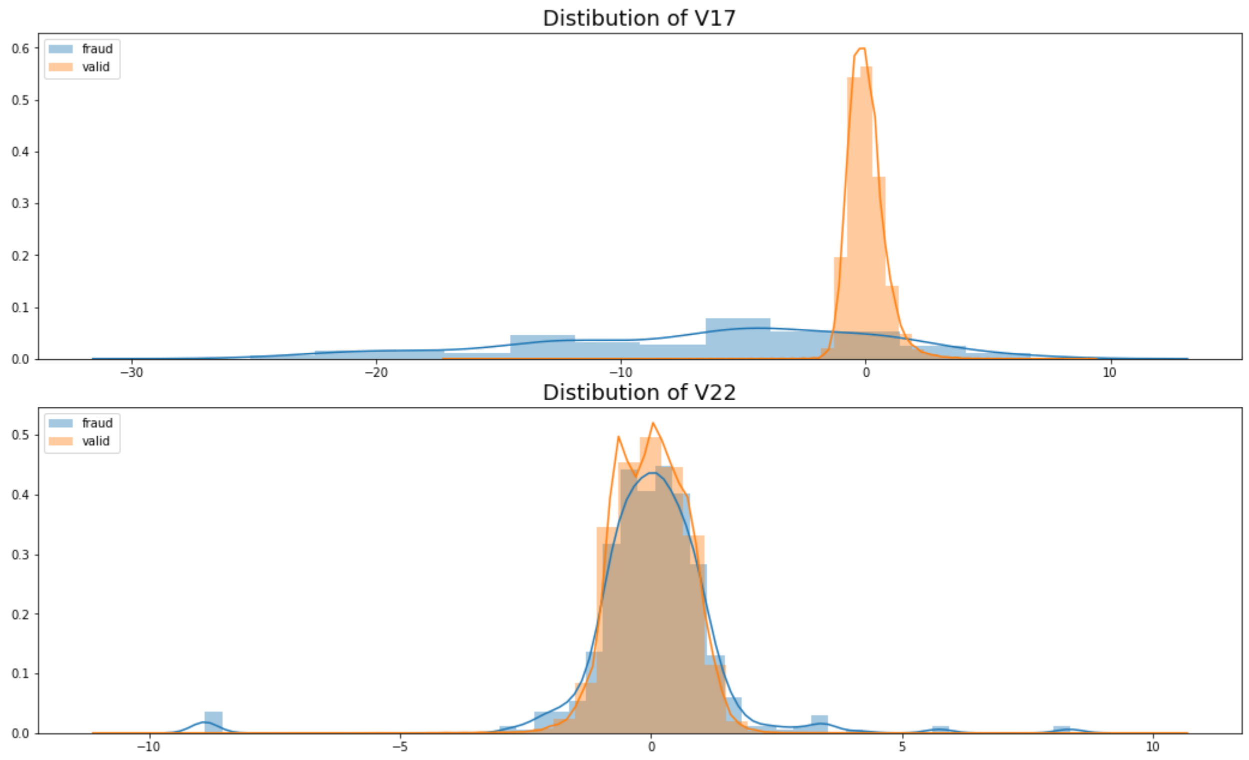


Figure 5: Examples of Distributions of Genuine and Fraudulent Transactions

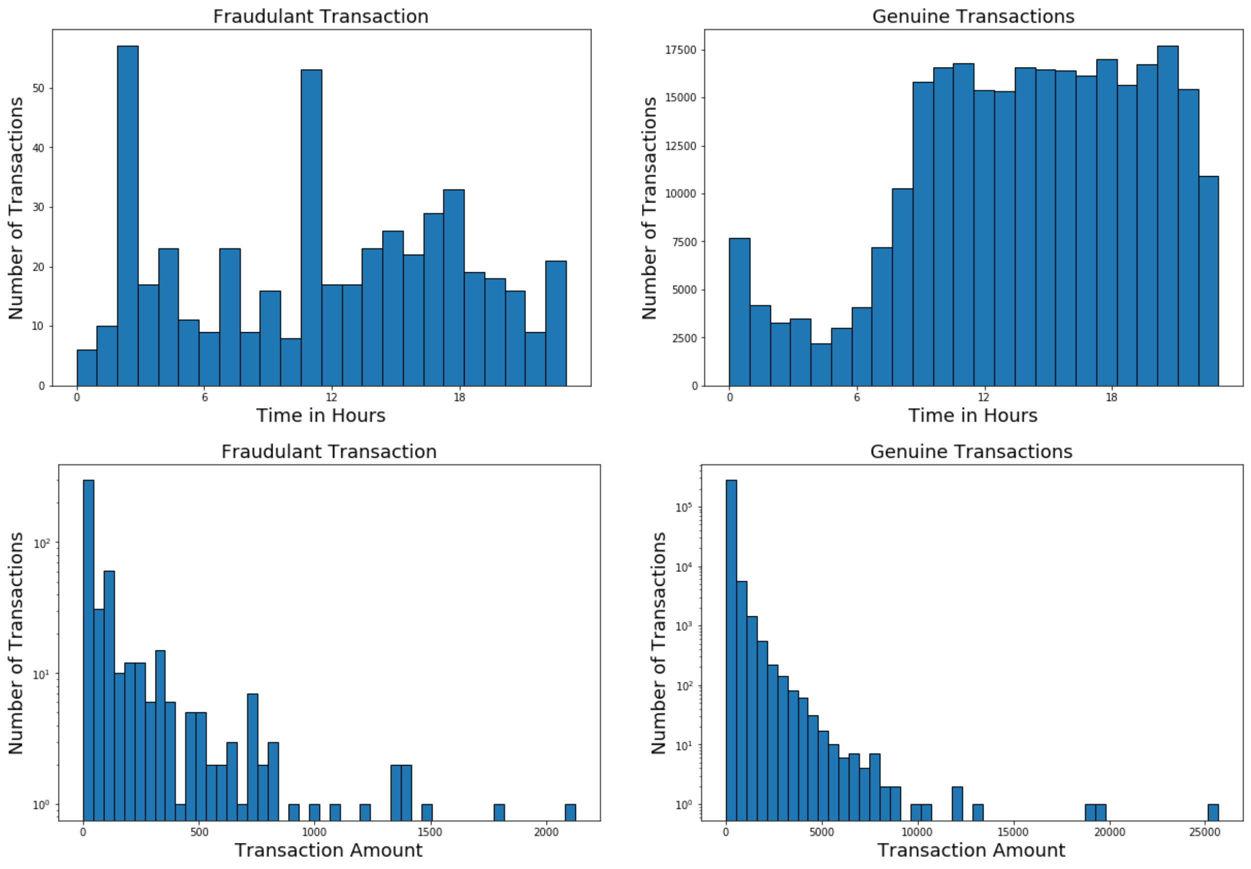


Besides V1-V28 features, the distributions of ‘Time’ and ‘Amount’ by class may provide useful information. However, they follow the same distributions in each class. The spikes in the fraudulent transactions Time feature distribution may result from the limited observations.

It is important to note that the dataset covers 2 days of transactions. In order to have a better visual, the ‘Time’ feature is consolidated and the ‘Amount’ feature is log-transformed. The distributions of ‘Time’ and ‘Amount’ variables are provided below.

It may be possible to select a subset of features to identify fraudulent transactions. However, this kind of an approach may come with its own limitations due to the lack of enough observations. For this reason, it is better to keep all features and train the machine learning algorithms with the most data we can find.

Figure 6: Histograms of Time and Amount



4. Default Models

In this section we will train and test the following models using the default hyperparameters.

- Logistic Regression
- Naive Bayes
- K-Nearest Neighbors Classifier
- Decision Tree Classifier
- Random Forest Classifier
- Extra Trees Classifier
- AdaBoost Classifier
- Gradient Boosting Classifier
- XGBoost Classifier

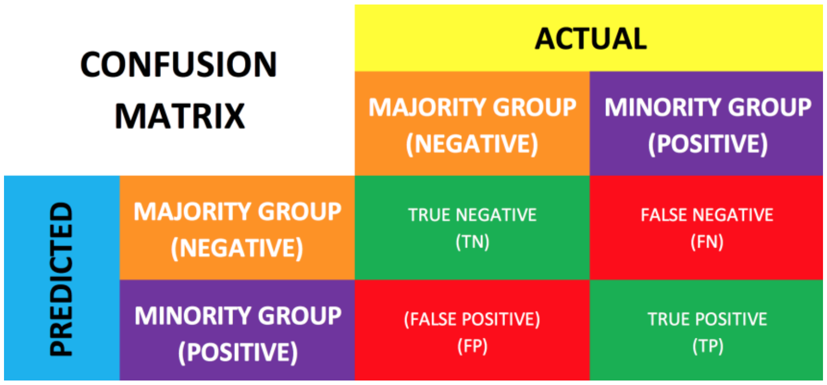
Before we move any further, it is a good practice to separate a subset of data as hold-out samples. Since we have a limited number of observations, splitting the dataset into training and test sets will provide more accurate performance of the machine learning algorithms. For this reason, we kept 30% of the data for testing purposes and the models are trained using 70% of the data. It is important to have approximately same class distribution both in training and test sets for the sake of the performance of the algorithms. Thus, we used Stratified 5-Fold cross validation to be able to sustain the class distribution.

The 5-Fold cross validation indicates that the training set will be split into 5 equal subsets and in each iteration one of the subsets will be kept for hold-out sample. However, it is important to train the models from scratch in each iteration in order to validate the cross-validation scores. Otherwise, the model will face the points in the test set that it has already faced in the training set which is called data leak.

4.1. The Metrics

Before diving into the world of metrics, it is a good idea to cover the confusion matrix. Confusion matrix is a summary of the performance of the classifier algorithm. For a binary classifier, the confusion matrix is constructed as follow:

Figure 7: The Confusion Matrix



CONFUSION MATRIX		ACTUAL	
		MAJORITY GROUP (NEGATIVE)	MINORITY GROUP (POSITIVE)
PREDICTED	MAJORITY GROUP (NEGATIVE)	TRUE NEGATIVE (TN)	FALSE NEGATIVE (FN)
	MINORITY GROUP (POSITIVE)	(FALSE POSITIVE) (FP)	TRUE POSITIVE (TP)

In the light of credit card fraud detection problem:

- True Negative (TN): Transactions that are genuine and classified as genuine.
- False Negative (FP): Transactions that are fraudulent but classified as genuine.
- True Positive (TP): Transactions that are fraudulent and classified as fraudulent.
- False Positive (FP): Transactions that are genuine but classified as fraudulent.

The most well-known classification metrics are:

- **Accuracy**: What percent of all transactions are classified correctly?

$$Accuracy = (TN + TP) \; / \; (TN + FN + FP + TP)$$

Accuracy Paradox: Accuracy scores from imbalanced data sets can lead to deceiving results. Since accuracy gives the percentage of correct classification, it is an easy job to reach 99% or above accuracy scores in an imbalanced data set. For example, in the credit card fraud detection data set, 99.83% of all transactions belong to the genuine transactions category which means that 99.83% accuracy score is guaranteed if all transactions are classified as genuine transactions. Besides, the machine learning algorithms will tend to categorize all transactions as genuine transactions due to the lack of enough observation in the fraudulent transactions' category. For this reason, it is important to avoid accuracy scores and to focus on identifying positive or minority class.

- **Precision**: What percent of transactions that are predicted to be fraudulent are classified correctly?

$$Precision = (TP) \; / \; (FP + TP)$$

- **Recall / Sensitivity**: What percent of (actual) fraudulent transactions are classified correctly?

$$Recall = (TP) / (FN + TP)$$

- **Specificity**: What percent of (actual) genuine transactions are classified correctly?

$$Specificity = (TN) / (TN + FP)$$

- **F1 Score / F2 Score**: (*Weighted*) harmonic average of **Recall** and **Precision**.

$$F1\,Score = (2 \times Recall \times Precision) / (Recall + Precision)$$

$$F2\,Score = (1 + 2^2) \times (Recall \times Precision) / (2^2 \times Precision + Recall)$$

- **Geometric Mean Score:** the square root of the product of the sensitivity and specificity.

$$Geometric\ Mean = \sqrt{Sensitivity \times Specificity}$$

- **Matthews Correlation Coefficient:** a measure of the quality of binary and multiclass classifications.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

  As will be seen in the future parts of the study, the MCC is one of the best metrics for imbalanced classification.

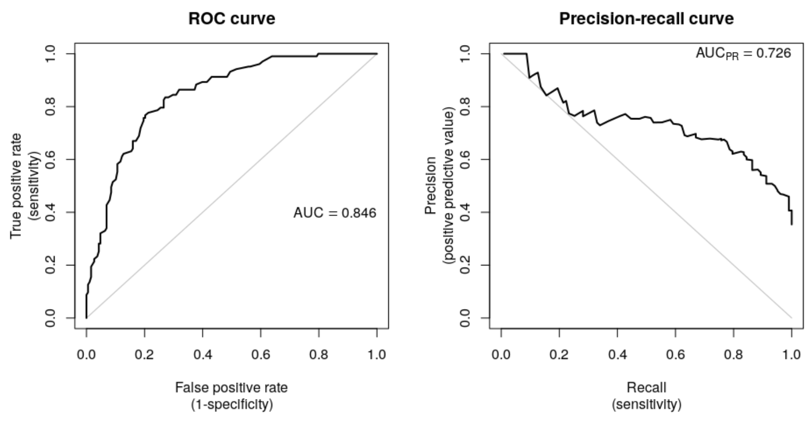
- **Precision-Recall Curve**

The plot of **Precision** on the y-axis and **Recall** on the x-axis.

- **ROC (Receiver Operating Characteristic) Curve**

The plot of **Sensitivity** (True Positive Rate) against **1 - Specificity** (False Positive Rate).

Figure 8: Receiver Operating Characteristic and Precision-Recall Curves



Plots from Precision-Recall and Receiver Operating Characteristic curves can be created and used to understand the trade-off in performance for different threshold values when interpreting probabilistic predictions. Each plot can also be summarized with an area under the curve (AUC) score that can be used to directly compare classification models (Brownlee, 2020).

For further information about the above metrics please refer to the documentation of Scikit-Learn and Imbalanced-Learn API documentations.

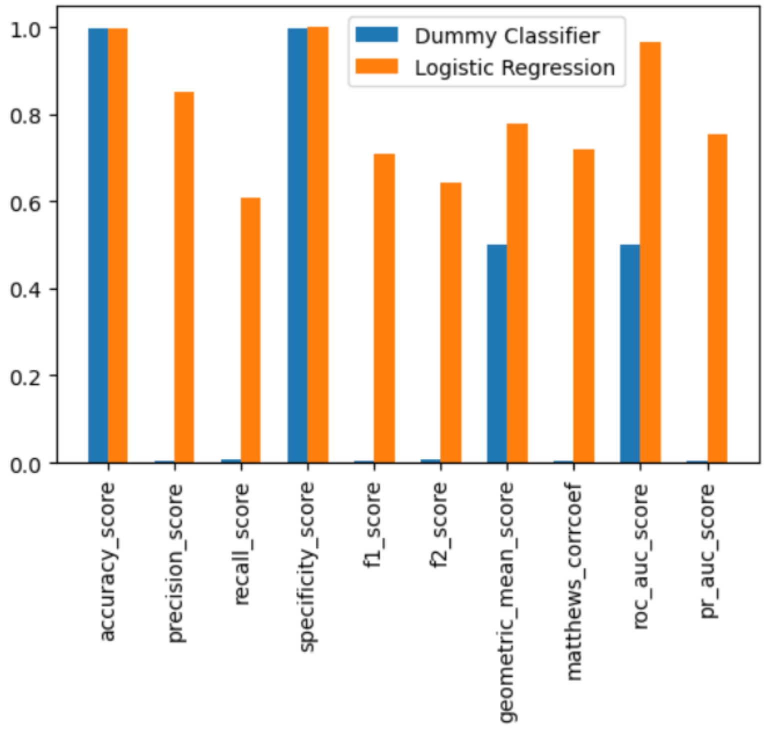
4.2. Dummy Classifier

Another important part of the cross-validation processes is to have a benchmark model which can distinguish the working models from the 'junk' models. For this reason, we will employ 'Dummy Classifier'.

The Dummy Classifier is a model which does not consider the input matrix while predicting the labels. Furthermore, having Dummy Classifier as a benchmark will save us from making investment on a 'junk' model. If a model is not able to beat the Dummy Classifier scores, then that model has worse predictive power than a model (Dummy Classifier) which assigns labels randomly. For this reason, we exclude any model which has lower scores than Dummy Classifier. Also, it will be a benchmark to decide the fruitful metrics.

We will use Logistic Regression and Dummy Classifier to decide which metrics are the most useful for the dataset. In Figure 9, the scores for each classifier is provided.

Figure 9: Initial Evaluation of Metrics



In the above figure we see the performance of Dummy Classifier and Logistic Regression for several evaluation metrics. The first thing is that **Accuracy** and **Specificity** are not good evaluation metrics, because Dummy Classifier can perform nearly the perfect score, as well as Logistic Regression. Because of the extremely imbalanced class distribution, the Dummy Classifier assigns all transactions to the negative class. Thus, it naturally reaches a score of 99.83% (the percentage of negative class).

The second thing is that the gap between **Precision** and **Recall**. While the former tells us 'What percent of predicted fraudulent transactions are classified correctly?'; the latter tells us 'What percent of (actual) fraudulent transactions are classified correctly?'. Even though, they are both concern about the fraudulent transactions, we would like to put more weight on **Recall** since it is an important task to catch as much fraudulent transactions as possible.

The third thing is that we observe the above pattern in the hybrid metrics such as **F1 Score**, **F2 Score** and **Geometric Mean**. The first two hybrid metrics (**F1/F2 Scores**) are based on **Precision** and **Recall** and the last one (**Geometric Mean**) is based on **Specificity** and **Sensitivity** (**Recall**). Since **Specificity** has a higher score than **Precision**, the **Geometric Mean** is higher than both of **F1/F2 Scores**.

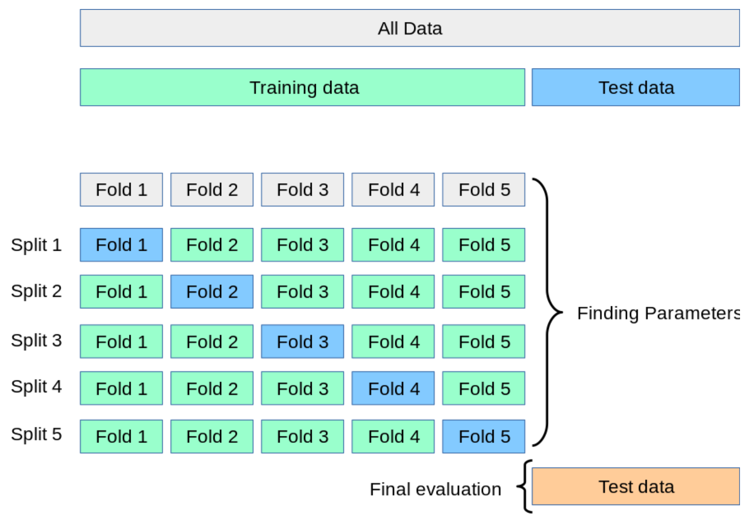
As a result, when the nature of data (being extremely imbalanced) and the above discussion points are taken into consideration, for now, we believe that it is best to continue with **Matthews CC**, **AUCPR** and **AUCROC**.

4.3. Cross-Validation Results

Cross-Validation lets us to reduce the impact of *luck* in our studies by training and testing the models from scratch with different parts of data and getting an interval for the future results.

In the process of cross-validation, the data is split into *k* subsets. In each iteration the algorithm is trained with (*k-1*) subsets and tested with the hold-out sample. In the study, we used a Stratified 5-Fold cross-validation and shuffled the data before each split in order to eliminate the impact of time since the data is presented as a time-series data, ordered by time it occurred. The cross-validation process is described visually in Figure 10.

Figure 10: Cross-Validation Process



In the table below, the cross-validation scores and the standard deviations are provided. Also, in Figure 11, the information is presented in bar graphs.

As seen in the below graph, Matthews CC and AUCPR scores are pretty close to each other regardless of the algorithm. On the other hand, all algorithms have almost perfect AUCROC score which is kind of suspicious when Matthews CC and AUCPR scores are taken into consideration. Also, Gradient Boosting Classifier has the highest error which means its scores varies in a large interval but it is not a desired outcome. Moreover, Naive Bayes has one of the highest AUCROC scores but it does not consistent with Matthews CC and AUCPR when compared with other models. Finally, as discussed in Brownlee (2020) ROC score may report optimistic results under the condition of extreme class imbalance. In this study, we believe that AUCROC scores reflects that optimism so that the highest score for each model is AUCROC score.

In Figure 12 and 13, we provided Precision-Recall and ROC curves for each iteration and the mean curve for the cross-validation. The above pattern can be observed by comparing two curves.

As a result, both Naive Bayes and Gradient Boosting Classifier are discarded from the algorithm set. Also, we will consider to exclude AUCROC from the metrics set; however, it is important to see the test results and the confusion matrices before taking this step.

Table 1: Cross-Validation Scores

	MCC Mean	**AUCROC Mean**	**AUCPR Mean**	**MCC Std**	**AUCROC Std**	**AUCPR Std**
Logistic Regression	0.719282	0.967877	0.755891	0.028513	0.012395	0.028631
Naive Bayes	0.298440	0.965817	0.331273	0.021472	0.009326	0.036603
KNN Classifier	0.824785	0.919728	0.852010	0.023925	0.022877	0.033320
Decision Tree Classifier	0.744170	0.881013	0.745941	0.055447	0.040143	0.054938
Random Forest Classifier	0.859027	0.943637	0.854041	0.030563	0.011255	0.028008
Extra Trees Classifier	0.850278	0.956764	0.859419	0.027048	0.013798	0.032587
AdaBoost Classifier	0.732488	0.969976	0.760508	0.046935	0.010269	0.051645
Gradient Boosting Classifier	0.661459	0.746732	0.602662	0.174120	0.201380	0.197956
XGBoost Classifier	0.848954	0.975323	0.850352	0.040524	0.004999	0.028412

Figure 11: Cross-Validation Scores

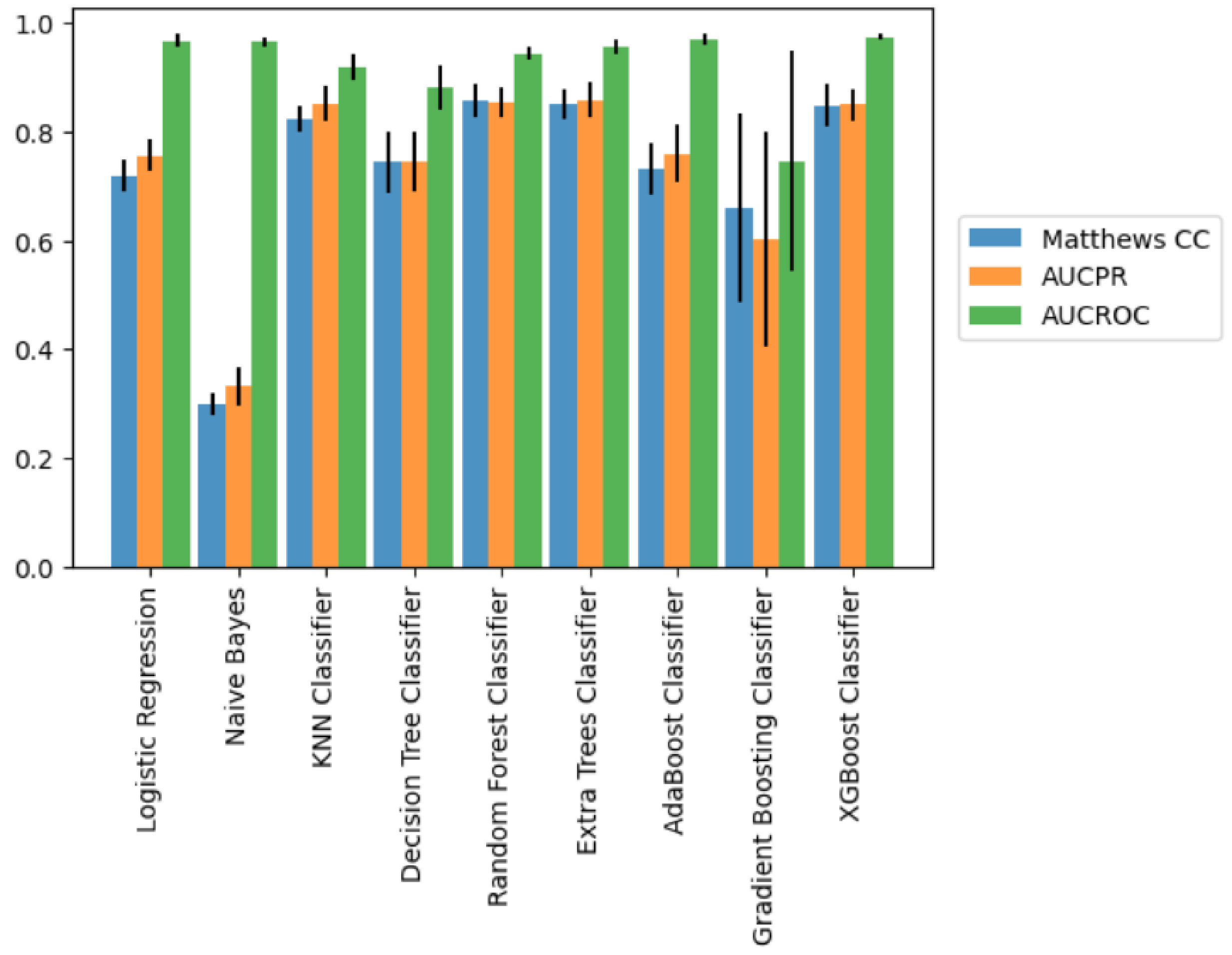
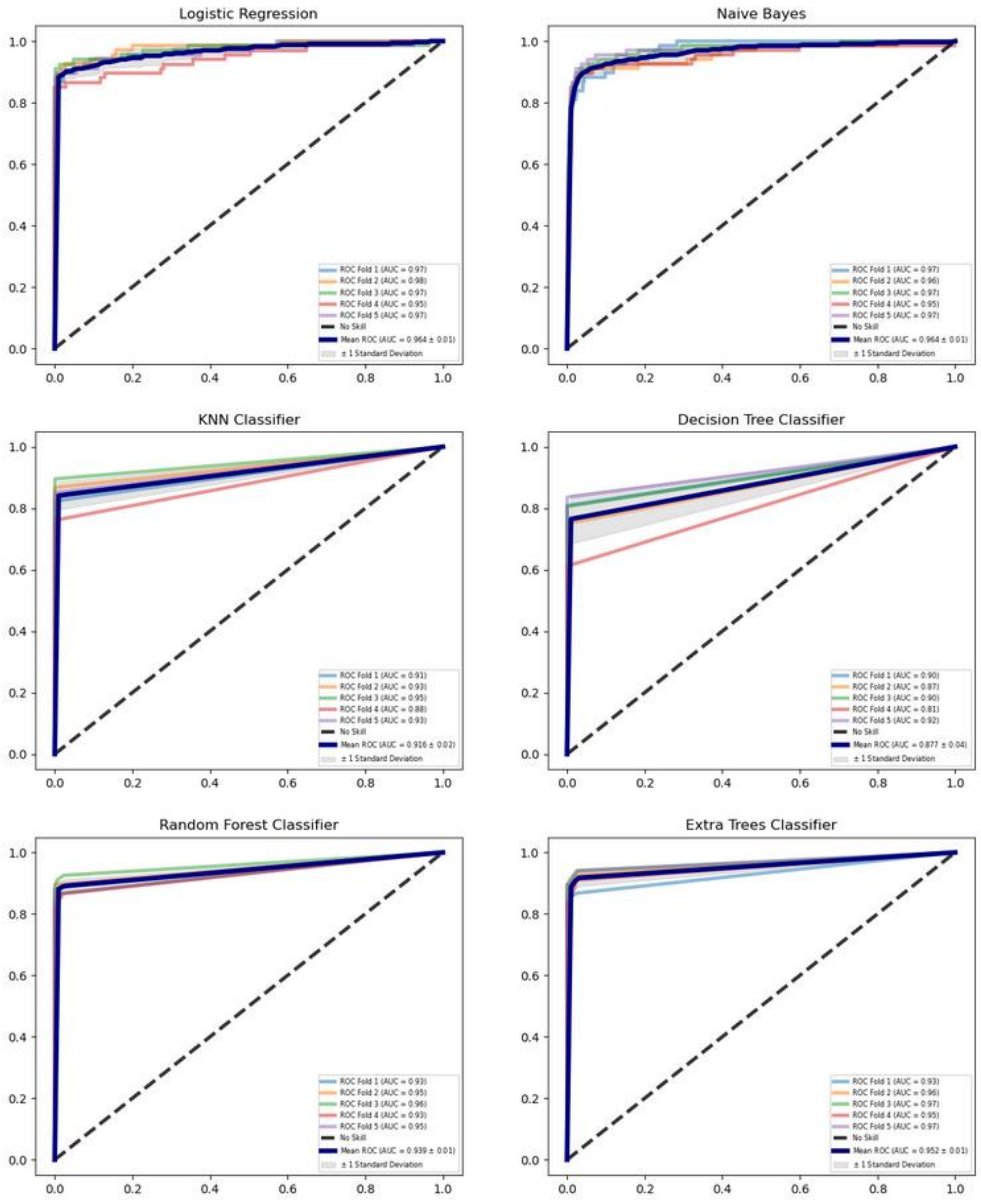


Figure 12: Receiver Operating Characteristic Curves



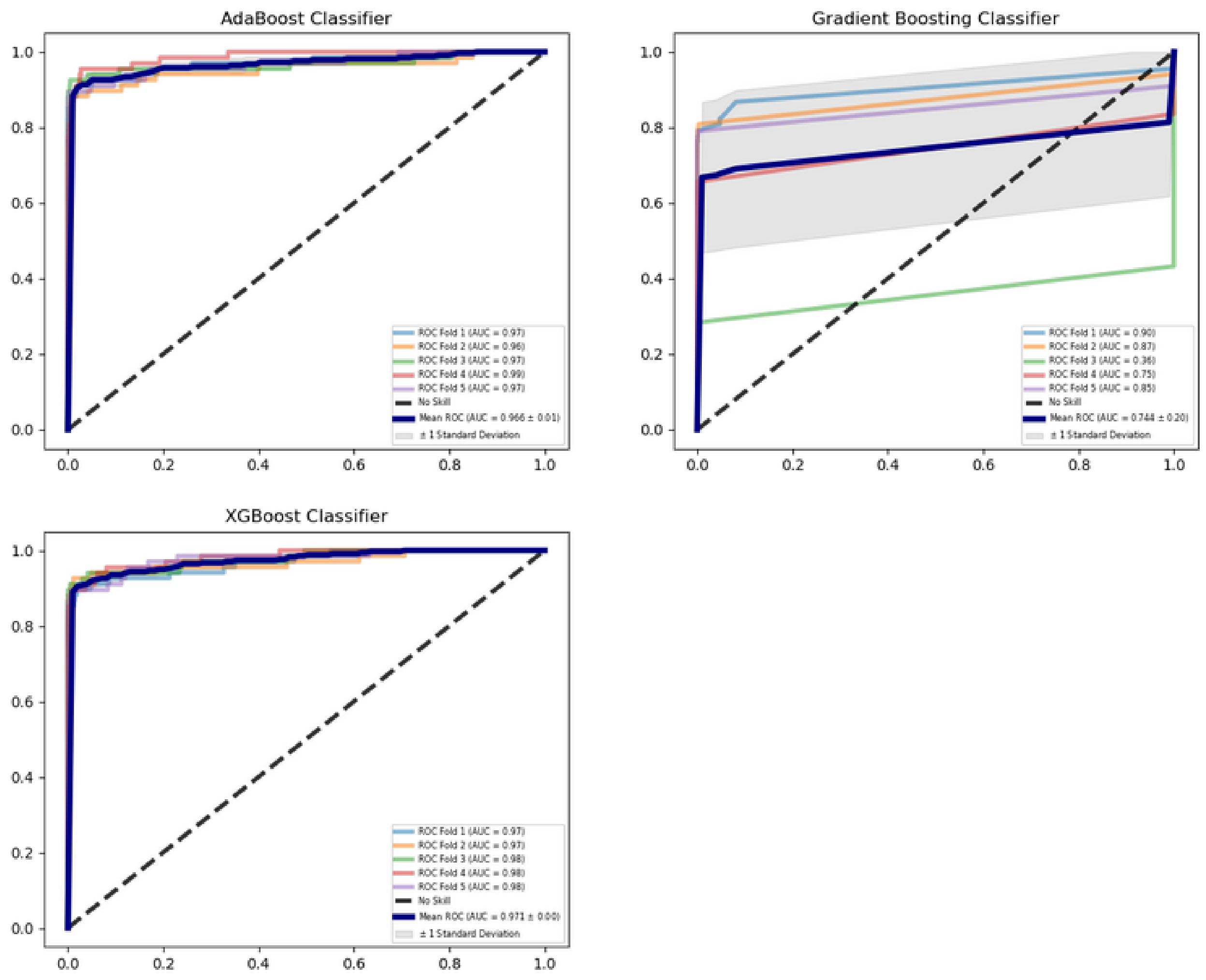
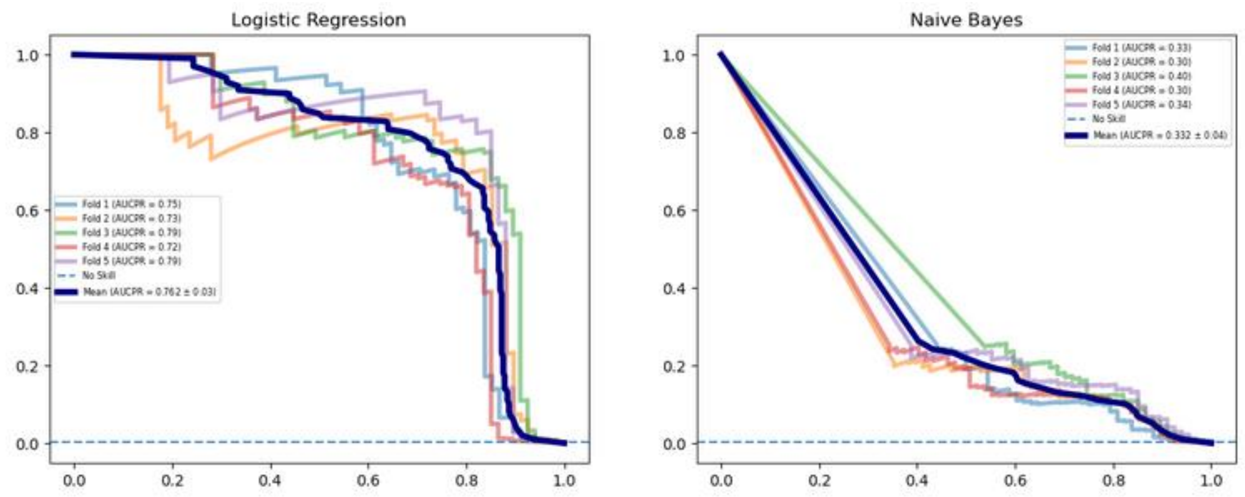
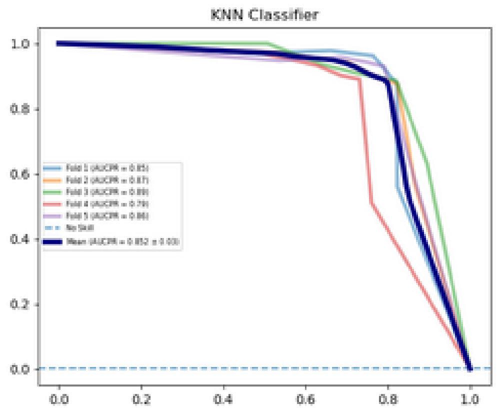
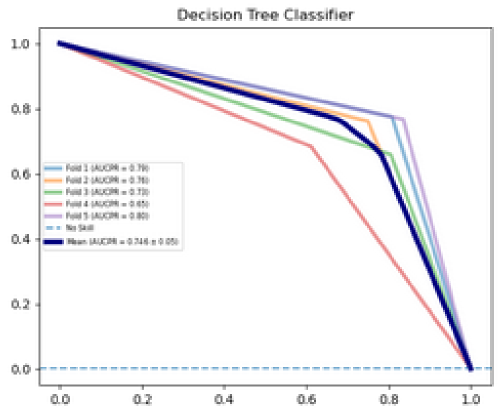
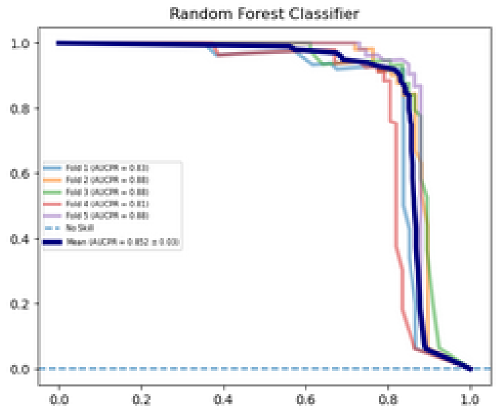


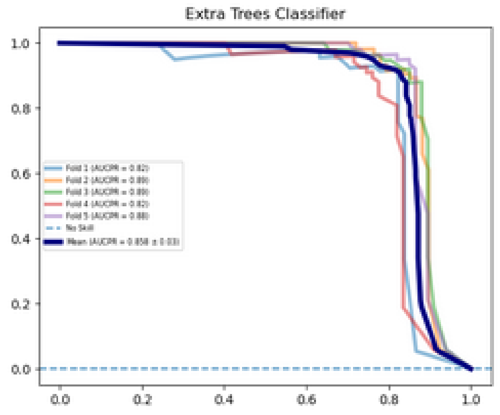
Figure 13: Precision-Recall Curves

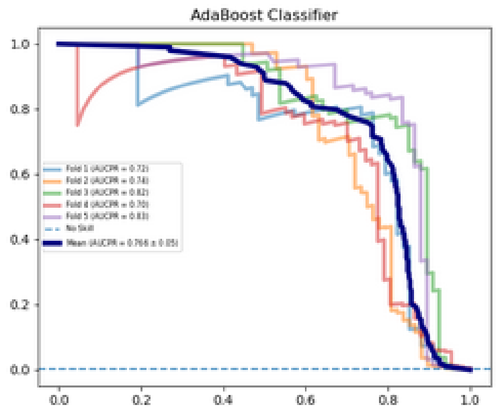


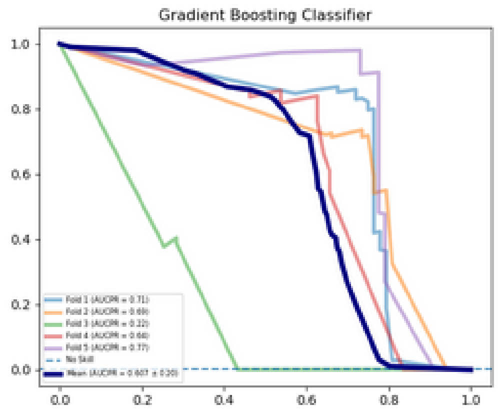

KNN Classifier
Fold 1 (AUCPR = 0.85)
Fold 2 (AUCPR = 0.87)
Fold 3 (AUCPR = 0.89)
Fold 4 (AUCPR = 0.79)
Fold 5 (AUCPR = 0.86)
No Skill
Mean (AUCPR = 0.852 ± 0.03)
0.0
0.2
0.4
0.6
0.8
1.0

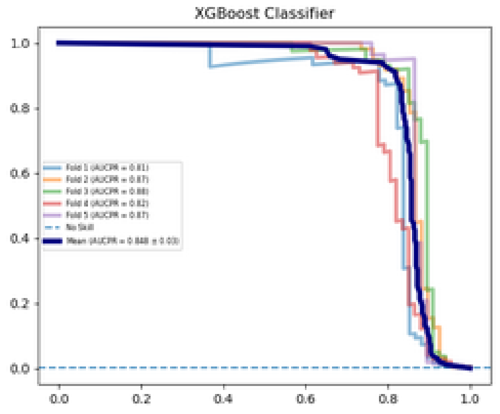

Decision Tree Classifier
Fold 1 (AUCPR = 0.79)
Fold 2 (AUCPR = 0.76)
Fold 3 (AUCPR = 0.73)
Fold 4 (AUCPR = 0.65)
Fold 5 (AUCPR = 0.80)
No Skill
Mean (AUCPR = 0.746 ± 0.05)
0.0
0.2
0.4
0.6
0.8
1.0


Random Forest Classifier
Fold 1 (AUCPR = 0.83)
Fold 2 (AUCPR = 0.88)
Fold 3 (AUCPR = 0.88)
Fold 4 (AUCPR = 0.81)
Fold 5 (AUCPR = 0.88)
No Skill
Mean (AUCPR = 0.852 ± 0.03)
0.0
0.2
0.4
0.6
0.8
1.0


Extra Trees Classifier
Fold 1 (AUCPR = 0.82)
Fold 2 (AUCPR = 0.89)
Fold 3 (AUCPR = 0.89)
Fold 4 (AUCPR = 0.82)
Fold 5 (AUCPR = 0.88)
No Skill
Mean (AUCPR = 0.858 ± 0.03)
0.0
0.2
0.4
0.6
0.8
1.0


AdaBoost Classifier
Fold 1 (AUCPR = 0.72)
Fold 2 (AUCPR = 0.74)
Fold 3 (AUCPR = 0.82)
Fold 4 (AUCPR = 0.70)
Fold 5 (AUCPR = 0.83)
No Skill
Mean (AUCPR = 0.766 ± 0.05)
0.0
0.2
0.4
0.6
0.8
1.0


Gradient Boosting Classifier
Fold 1 (AUCPR = 0.71)
Fold 2 (AUCPR = 0.69)
Fold 3 (AUCPR = 0.22)
Fold 4 (AUCPR = 0.64)
Fold 5 (AUCPR = 0.77)
No Skill
Mean (AUCPR = 0.607 ± 0.20)
0.0
0.2
0.4
0.6
0.8
1.0


XGBoost Classifier
Fold 1 (AUCPR = 0.81)
Fold 2 (AUCPR = 0.87)
Fold 3 (AUCPR = 0.88)
Fold 4 (AUCPR = 0.82)
Fold 5 (AUCPR = 0.87)
No Skill
Mean (AUCPR = 0.848 ± 0.03)
0.0
0.2
0.4
0.6
0.8
1.0

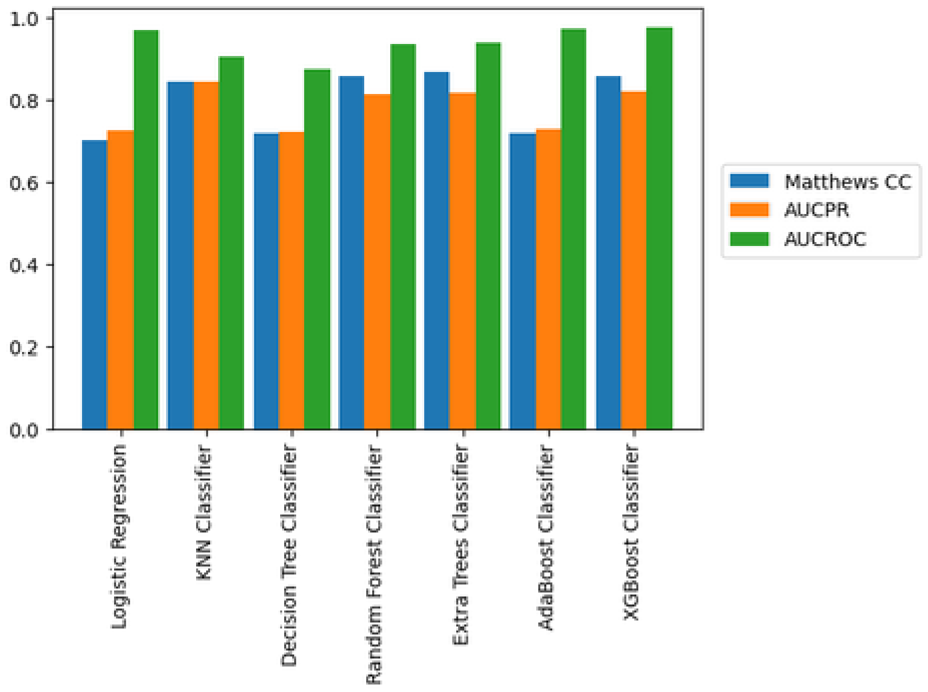
4.4. Test Scores and Confusion Matrices

After the cross-validation step, algorithms are trained with the whole training data (70%) and are tested with test data. In Table 2, we provided the test scores for the machine learning algorithms. Also, in Figure 14, we provided the bar graph for visual understanding of the test scores.

Table 2: Test Scores

	MCC	AUCROC	AUCPR
Logistic Regression	0.701298	0.967966	0.724175
KNN Classifier	0.845124	0.906128	0.843538
Decision Tree Classifier	0.719272	0.874746	0.720512
Random Forest Classifier	0.857865	0.935696	0.812826
Extra Trees Classifier	0.866544	0.939164	0.816161
AdaBoost Classifier	0.719188	0.972236	0.727773
XGBoost Classifier	0.857487	0.976248	0.819320

Figure 14: Test Scores

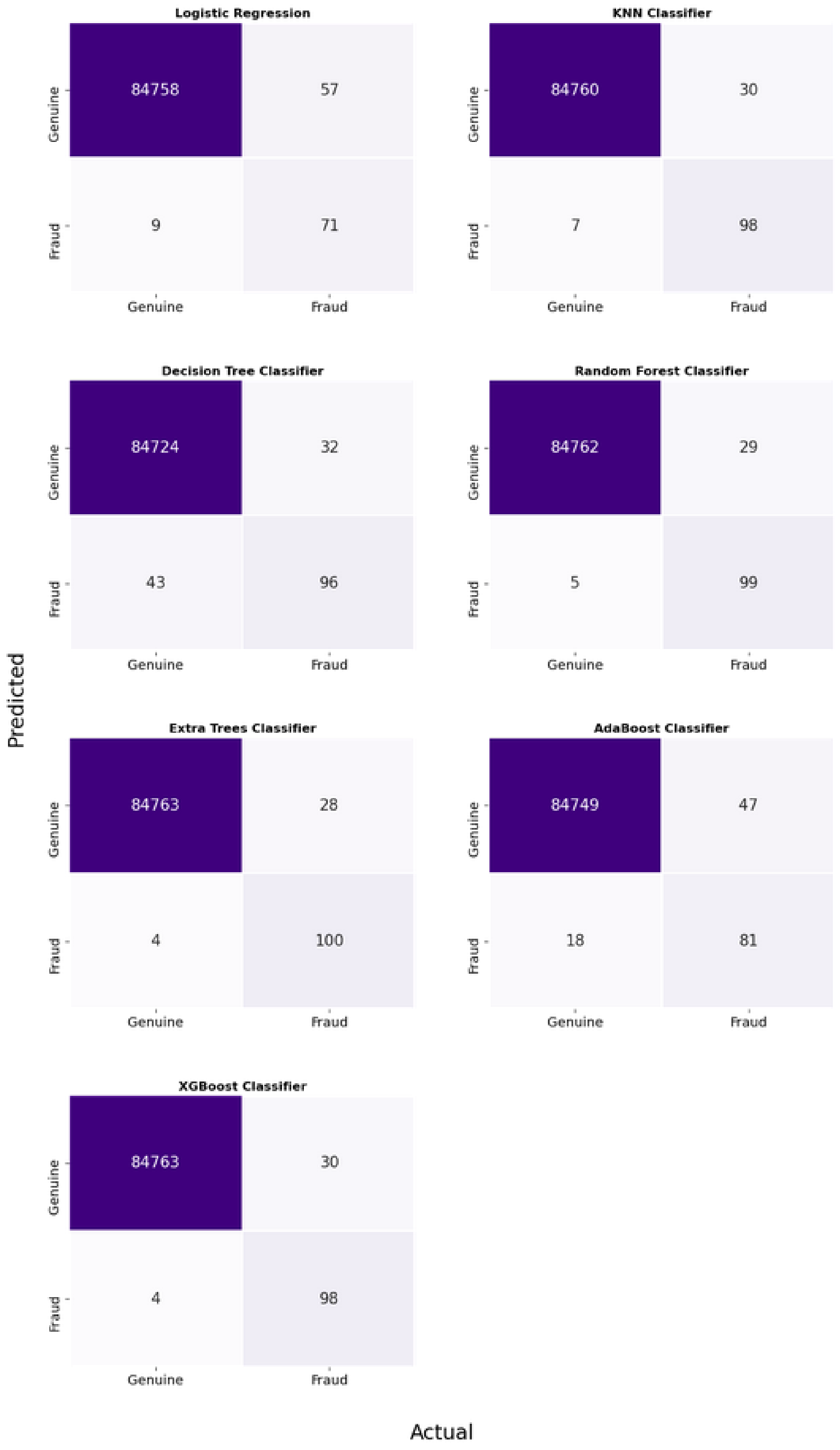


In the confusion matrices, in Figure 15, it is possible to see the quality of the classification can be seen. It is also color-coded which means that the greater the number of transactions in a square the darker the color is.

The right bottom square gives the number of true positives which are the correctly classified fraudulent transactions. Also, the left upper square gives the number of true negatives which are the correctly classified genuine transactions. For this reason, we will look for the left upper and right bottom squares to have the greatest number of transactions but the other two squares to have nearly 0 transactions.

It is seen that the following are the best (in order of performance): Extra Tree Classifier, XGBoost Classifier, Random Forest Classifier, and KNN Classifier.

Figure 15: Confusion Matrices



4.5. Sampling Methods

In order to address the imbalanced class distribution of the dataset, we employed SMOTEENN - Synthetic Minority Over-sampling Technique and Edited Nearest Neighbours, sampling technique. It is possible to use an under-sampling, or an over-sampling to balance the class distribution. However, an under-sampling method will reduce the number of genuine transactions by deleting the valid transactions. This kind of an approach will not benefit the study since we will lose legit data. On the other hand, an over-sampling technique will increase the number of fraudulent transactions by generating synthetic data. This kind of an approach will benefit the study since the pattern with the fraudulent transactions will be easily recognizable. However, the quality of the boundary between the classes is as much important as the class distribution. For this reason, we used a hybrid approach, SMOTEENN, by generating synthetic data for fraudulent transactions, then deleting confusion points in the boundary in order to increase the quality of classification.

On the other hand, we experienced the following sampling methods before we come to final decision:

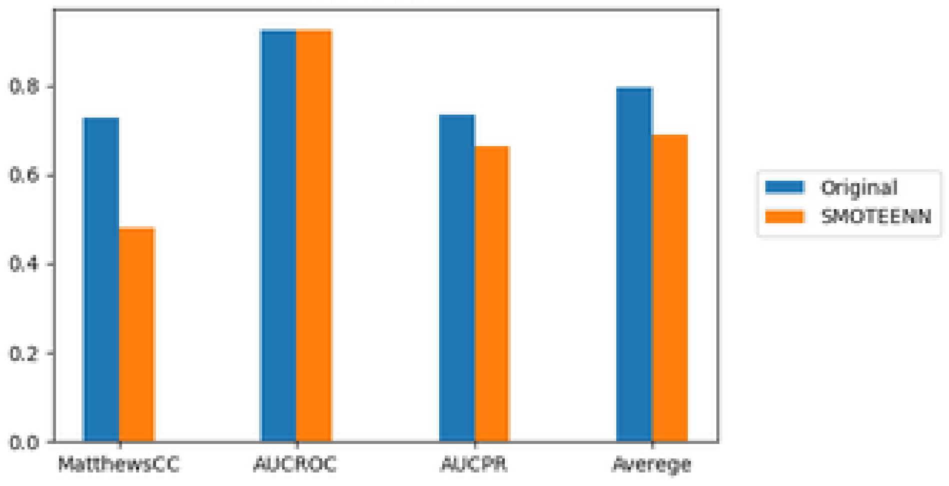
- One Sided Selection - OSS (Under-sampling)
- Neighborhood Cleaning Rule - NCR (Under-sampling)
- Synthetic Minority Oversampling Technique - SMOTE (Over-sampling)
- Synthetic Minority Oversampling Technique and Edited Nearest Neighbours – SMOTEENN
- Synthetic Minority Oversampling Technique with Tomek's Links - SMOTETomek

The best performing algorithm is SMOTEENN for the dataset in hand. It is possible to find detailed information about sampling methods on Imbalanced-Learn API documentation.

In Figure 16, the mean scores for the metrics for data with SMOTEENN and without the resampling method is plotted. The mean scores are the average of the scores of all algorithms for each metric. The reasoning is that an increase in the quality of the data will be visible on the metrics so that SMOTEENN will generate higher scores on average than the original dataset.

As seen in the figure, SMOTEENN did not improve the test scores. For this reason, it is better to continue with the original dataset.

Figure 16: Mean Scores for Metrics with SMOTEENN



5. Tuning the Default Models

In the hyperparameter tuning process, we used Stratified 5-Fold cross-validation with GridSearchCV. The cross-validation process shows us the performance of an algorithm for different parts of the data which eliminates the *luck*. The GridSearchCV goes over every parameter in the parameter space, and trains and tests the algorithm from scrap in each iteration. It may require a great amount of time depending on the size of the hyperparameter space, the characteristic of the algorithm, and the size of the dataset.

After identifying the algorithms that lead to best cross-validation score, the next step is to train and test the algorithms with the new set of parameters. Because of the limited computing power, we focused on the most important arguments for each algorithm. The list of arguments for each algorithm is given below:

- Logistic Regression: **solver, C, class_weight**
- K-Nearest Neighbor Classifier: **n_neigbors**
- Decision Tree Classifier: **max_features, class_weight**
- Random Forest Classifier: **n_estimators, max_features, class_weight**
- Extra Trees Classifier: **n_estimators, max_features, class_weight**
- AdaBoost Classifier: **n_estimators, learning_rate**
- Gradient Boosting Classifier: **n_estimators, learning_rate, max_features**
- XGBoost Classifier: **eta, colsample_bytree**

In Table 4, we presented AUCPR scores with default parameters and the optimized parameters.

Table 3: AUCPR Score with Default and Optimized Parameters

	Default Parameters	Tuned Parameters	Change
Logistic Regression	0.756	0.758	0.002
Decision Tree Classifier	0.734	0.761	0.027
Random Forest Classifier	0.854	0.860	0.060
Extra Trees Classifier	0.861	0.864	0.003
AdaBoost Classifier	0.761	0.829	0.068
Gradient Boosting Classifier	0.604	0.828	0.224
XGBoost Classifier	0.850	0.861	0.011
K-Nearest Neighbors Classifier	0.852	0.859	0.007

After completing the hyperparameter optimization, we can move to the final evaluation of the algorithms. In the final step, we will train the algorithms with the best parameters using the whole training data. Finally, we can test the algorithms and compare the test scores and confusion matrices with the default algorithm counterparts to find the best fit for the dataset in hand.

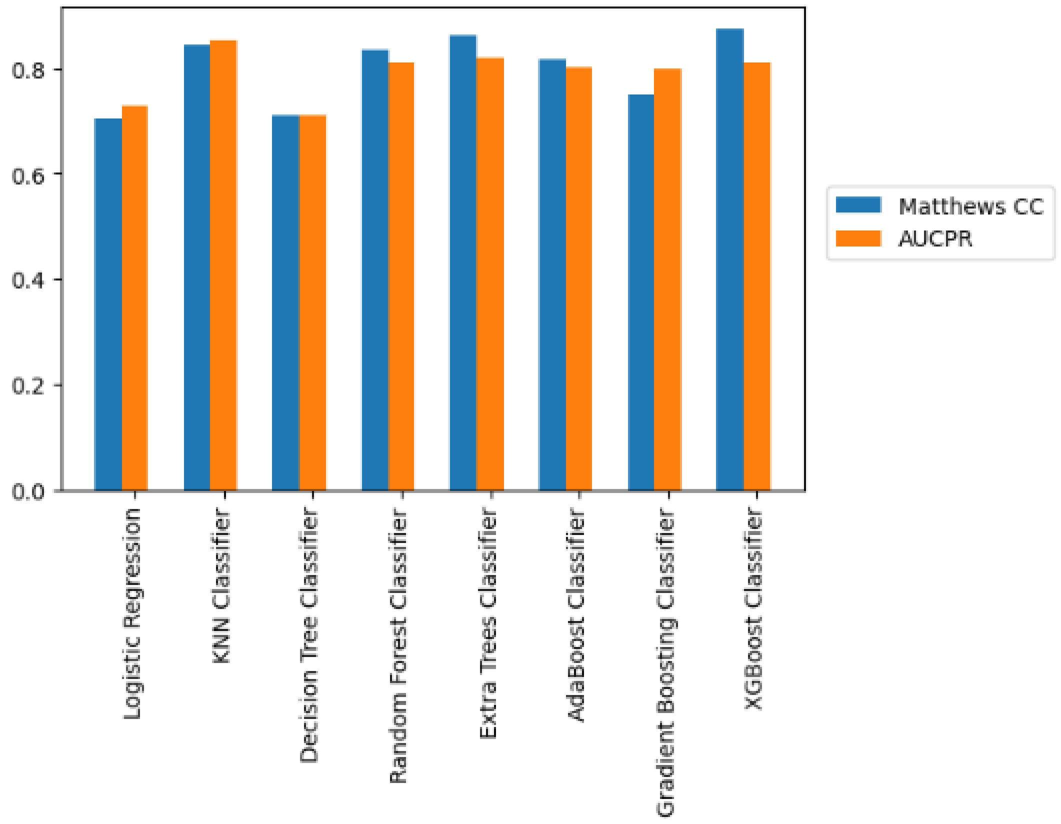
6. Final Evaluation

In Table 3, we reported the test scores including Gradient Boosting Classifier. In the optimization stage, Gradient Boosting Classifier increased its score by 0.22 points. For this reason, we would like to see how it will perform with the optimized parameters. In Figures 17 and 18, we provided test scores for visual inspection and the confusion matrices with optimizes parameters, respectively.

Table 4: Test Scores with Optimized Parameters

	MCC	AUCPR
Logistic Regression	0.705733	0.729675
KNN Classifier	0.845124	0.854349
Decision Tree Classifier	0.710030	0.710732
Random Forest Classifier	0.834420	0.811859
Extra Trees Classifier	0.861730	0.821606
AdaBoost Classifier	0.817036	0.803204
Gradient Boosting Classifier	0.751309	0.800484
XGBoost Classifier	0.875223	0.812641

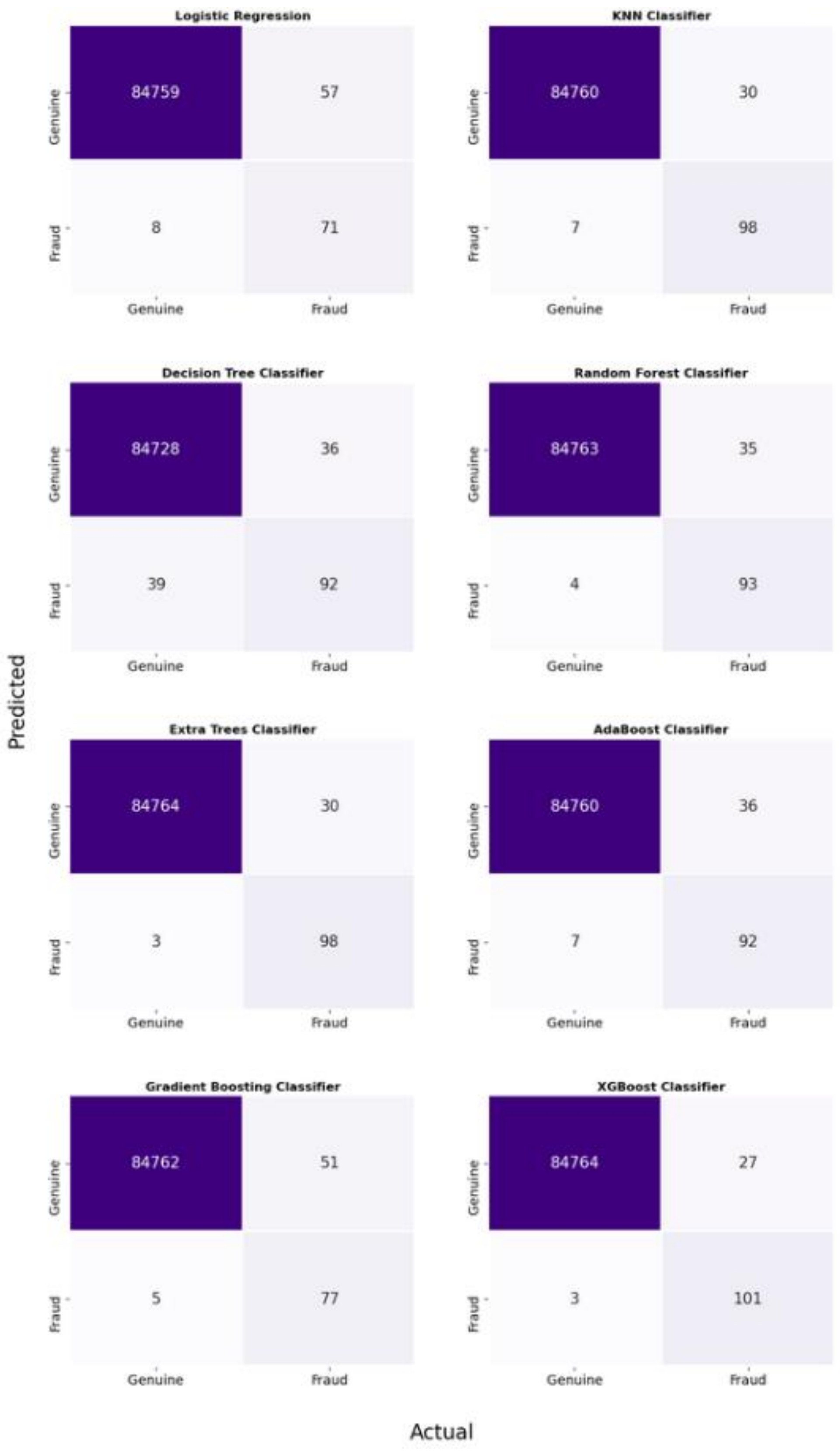
Figure 17: Test Scores with Optimized Parameters



In the Hyperparameter Optimization, a narrow set of hyperparameters were tested and the best performing pair of parameters are selected to be tested for the final evaluation. Even though, Gradient Boosting Classifier with default parameters is dropped from the study, it is included in the hyperparameter optimization and did a great job by increasing AUCPR score from 60% to 82%. For this reason, it is included in the final evaluation.

In the final stage, eight algorithms are tested, and the evaluation metric scores and the confusion matrices are reported. There is not a single pattern in the results. While Decision Tree Classifier, Random Forest Classifier and Extra Trees Classifier increased AUCPR score, they actually did not outperform their default model performances. For example, Decision Tree Classifier correctly predicted 97 out of 128 fraudulent transactions with the default parameters. However, after tuning the parameters, it was only able to correctly predicted 87 out of 128 fraudulent transactions.

Figure 18: Confusion Matrices with Optimized Parameters



On the other hand, XGBoost Classifier slightly improved the AUCPR score and the improvement is also observable in the confusion matrix scores. Also, AdaBoost Classifier improved its performance substantially. However, the improvement is not good enough to be the top algorithm for fraud detection.

Logistic Regression and K-Nearest Neighbor Classifier stayed stable and it can be observed from AUCPR scores of default models and tuned models. Hyperparameter optimization, overall, did have a positive effect on the performance of algorithms. Based on the test scores and the confusion matrices the top 3 algorithms are:

- XGBoost Classifier (Tuned Parameters)
- Extra Trees Classifier (Default Parameters)
- K-Nearest Neighbors (Default Parameters)

Finally, the best algorithm is the XGBoost Classifier. Moreover, Matthews CC is as good as AUCPR. Actually, with the tuned algorithms, it is a better indicator of the performance than AUCPR.

7. Conclusion

In this project we aimed to detect the fraudulent transactions from a 2-days of transactions. The dataset consists of 31 features in which 28 of them are the components acquired from PCA. The remaining three features are Time, Amount and Class.

In the preprocessing stage, we dropped the 0$ transactions from the dataset. Later, we separated data into input matrix and target matrix. The input matrix consists of all features except Class and the target matrix consists of Class feature.

For a comprehensive evaluation, we selected 10 evaluation metrics Accuracy, Precision, Recall, Specificity, F1 Score, F2 Score, Geometric Mean Score, Matthews Correlation Coefficient, AUC Precision-Recall Score and AUC ROC score. In the first evaluation of metrics, we compared the scores from Dummy Classifier with those from Logistic Regression. As a result, we decided to continue with Matthews Correlation Coefficient, AUC Precision-Recall and AUC ROC Scores.

In the cross-validation process, we used Stratified 5-Fold cross-validation and tested the following algorithms: Logistic Regression, K-Nearest Neighbors, Decision Tree Classifier, Naïve Bayes, Random Forest Classifier, Extra Trees Classifier, AdaBoost Classifier, Gradient Boosting Classifier and XGBoost Classifier. However, we dropped Naïve Bayes and Gradient Boosting Algorithm from the algorithm set due to unsatisfactory performances. Later on, we trained and tested the algorithms with default parameters. In the test scores, we find out that AUC ROC score behaves optimistically so that it was dropped from the set of metrics. After the training and testing with default parameters, we resampled the dataset, using SMOTEENN, in order to improve the class distribution. However, resampling did not improve the test scores.

Moreover, we optimized the parameters by using a narrow hyperparameter space and checked the cross-validation scores if the algorithms do better after optimization. There was not a single pattern for all algorithms. Gradient Boosting Classifier was the most successful algorithm and it improved the AUC Precision-Recall score by 0.22 points. For this reason, it is included to set of algorithms for the final evaluation.

In the final evaluation process, we trained the algorithms with optimized parameters using the whole training data and obtained the test scores for the hold-out sample. Based on the final test scores, XGBoost Classifier is the best classifier among all classifiers in terms of detecting the most fraudulent transactions and lowering the number of false negatives. It was able to detect 78.9% of all fraudulent transactions with only three false negatives.